1. IMPLEMENTACION

1.1 Implementación

El anexo del presente documento, contiene todo el código que corresponde el programa que aquí se intenta explicar .

El código que aquí se expone, está implementado de tal forma, que hay un código fuente que es el que va llamando a las correspondientes subrutinas.

Esta subrutina principal se ha llamado "**main_Fun1000**" , es la que controla todo el programa ,y es la que hay que ejecutar desde la ventana de comandos de MATLAB para iniciar el programa.

Al ejecutar "**main_Fun1000**" se crea una ventana que lee el número de materiales que componen la esquina y si esta es abierta o cerrada. Desde aquí se llama a la subrutina "**intromat**" (incluida en "**main_Fun90**"). Esta subrutina es la que lee el número de materiales que componen la esquina y en función de eso crea una nueva ventana preguntando sobre el tipo de material que es cada uno de ellos. Además, desde "**intronmat**" se llama a otras cuatro subrutinas: "**introMaterial**" "**calculate2**" "**calculate**" y "**getroots**".

La subrutina "**introMaterial**" es la que lee el tipo de material que es cada uno de los que componen la esquina y en función de eso llama a las siguientes subrutinas que cargan los datos del material. Estas son ,en función de si se trata de un material isótropo, ortótropo o transversalmente isótropo, "**pIso12(i)**" ," **pOrto4(i)**" y " **pTransIso(i)**" respectivamente. Estas subrutinas, están fuera del archivo principal y se guardan en archivos independientes que el código lee.

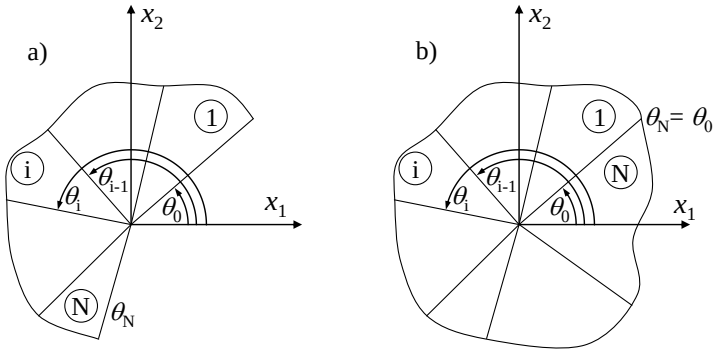
El archivo "**pIso12(i)**" crea una ventana que lee los datos de materiales isótropos y llama a su vez a la subrutina "**introE71**" que calcula la matriz de transferencia del material.

El archivo "**pOrto4(i)**" crea una ventana que lee los datos de materiales ortótropos y llama a su vez a la subrutina "**introE40**" que calcula la matriz de transferencia del material.

El archivo "**pTransIso(i)**" crea una ventana que lee los datos de materiales transversalmente isótropos y llama a su vez a la subrutina "**introEIso**" que calcula la matriz de transferencia del material.

Tanto las subrutinas "**pIso12(i)**" ," **pOrto4(i)**" y " **pTransIso(i)**" como las subrutinas "**introE71**", "**introE40**" y "**introEIso**" se encuentran guardadas en archivos independientes de la rutina principal, de forma que, si se quisiera añadir algún otro tipo de material a los cálculos, se podría hacer añadiendo una nueva subrutina, y sólo habría que llamarla desde "**introMaterial**".

Una vez que se ha ejecutado "**introMaterial**", la función principal guarda las matrices de transferencia de cada material y es la subrutina " **calculate2**" la que lee si la esquina es abierta o cerrada, y en función de eso, se ejecutan la subrutina "**Opened**" o "**Closed**". Dichas subrutinas también se encuentran fuera del archivo principal.



3-1 Esquinas abiertas y cerradas

El archivo "**Opened**" crea una ventana que pregunta sobre las condiciones de contorno e interfaz de la esquina, ejecuta a su vez la subrutina "**calcular3** " que llama a las subrutinas "**frictionless_o(i)**", "**bounded_o(i)**", "**Free_o(i)**", "**Fixed_o(i)**", "**Simetry_o(i)**", "**Antisimetry_o(i)**", " **Ur_restricted_o(i)**", "**Ur_allowed_o(i)**", "**U3_restricted_o(i)**" "**U3_allowed_o(i)**", "**BC_Friction_o(i)**" en función de las condiciones que se impongan. En el caso de las condiciones de contorno homogéneas las subrutinas correspondientes activan una ventana por cada condición de contorno en la que se introduce el ángulo inicial o final (según corresponda). En el caso de la condición de interfaz, pegado, deslizamiento y fricción, se introduce el ángulo que posiciona la interfaz, y en los casos de fricción se introduce, adicionalmente, el coeficiente de fricción. Estos datos se leen y se introducen en las correspondientes matrices de condiciones de contorno e interfaz en la siguiente subrutina que se ejecuta dentro de "**main_Fun1000**", que es la subrutina "**calculate"**.

La subrutina "**Closed**" crea una ventana que pregunta sobre las condiciones interfaz de la esquina y ejecuta a su vez la subrutina "**calculate3**" que llama a las rutinas "**frictionless(i)** "**,** "**friction(i)** ", "**bounded_i(i)**" en función de las condiciones que se impongan. En los casos de condición de interfaz de pegado, deslizamiento y fricción, se introduce el ángulo que posiciona la interfaz, y adicionalmente, en los casos de fricción se introduce el coeficiente de fricción. Estos datos se leen y se introducen en las correspondientes matrices de condiciones de contorno e interfaz en la siguiente subrutina que se ejecuta dentro de "**main_Fun1000**", que es la subrutina "**calculate"**.

La función "**calculate"** monta las matrices de condiciones de contorno e interfaz según corresponda, lee cuales son las matrices de transferencia de cada material, y monta la matriz de transferencia global del problema de la esquina multilateral en función de si esta es abierta o cerrada y de si hay condiciones de fricción o no. En el caso de que la matriz globla de trasferencia sea cuadrada (es decir no haya fricción), dentro de esta subrutina,se dibuja la parte real del determinante para valores de λ entre 0 y 1, para de esa manera identificar los ceros de la ecuación.

$$\det K_{corner}(\lambda) = 0$$

En el caso de que la matriz de transferencia global sea rectangular (incluya condiciones de fricción), dentro de la subrutina "**calculate**" se llamará a la subrutina "**procesador7**" que dibuja los valores de σmin para los distintos valores de λ entre 0 y 1, y ω entre -π y π, para de esa manera identificar los ceros de la ecuación.

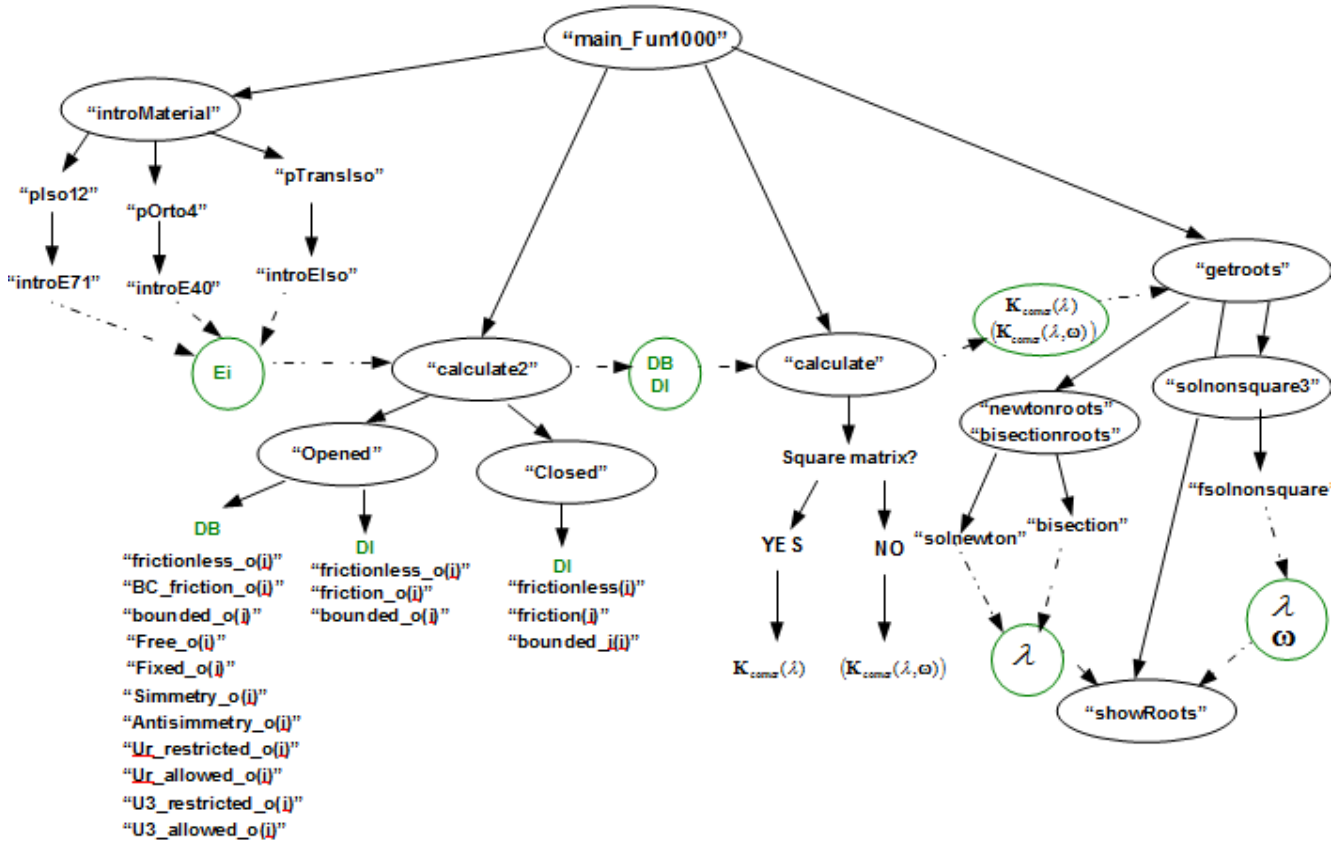
La subrutina "**getroots**" es la que calcula las raíces en función de si estamos en casos en los que exista fricción o no.

En los casos en los que no exista fricción, se podrán ejecutar las subrutinas "**newtonroots**" o "**bisecroots**" en función de si se quiere usar el método de Newton o el de la bisección para calcular las raíces del problema dentro del intervalo real de 0 a 1. Dichas subrutinas crean unas ventanas en las que se introducen los valores cerca de los cuales se encuentran las soluciones del problema, y mediante las subrutinas "**solnewton**" y "**bisection**" calculan las raíces del problema y las almacenan para mostrarlas posteriormente .

En los casos en los que exista fricción, se podrá ejecutar la subrutina "**solnonsquare3**" que utiliza el algoritmo de Levenberg-Marquardt para calcular las raíces. Esta subrutina crea unas ventanas en las que se introducen los valores de λ entre 0 y 1, y ω entre -π y π, cerca de los cuales se encuentran las soluciones del problema, y mediante las subrutinas "**fsolnonsquare3**" calcula las raíces del problema y las almacenan para mostrarlas posteriormente.

También, dentro de "**getroots**", se llama a la subrutina "**showRoots**" que permite visualizar las raíces por pantalla, y escribe el archivo de texto "**Roots.txt**" en el que se almacenan las raíces solución del problema de la esquina. Además, esta función, borra todos los parámetros anteriores usados durante el cálculo del problema, lo que permite el uso del programa nuevamente, sin problemas de interferencias entre variables o en sus resultados.

En la siguiente figura se muestra un diagrama de flujos para intentar clarificar el proceso que se sigue en el programa:



Figu 3-2 Esquema del programa